

[AWS Security Blog](#)

# How to Delegate Management of Multi-Factor Authentication to AWS IAM Users

by Mike Kuentz | on 02 JUN 2015 | in [AWS Identity And Access Management \(IAM\)](#), [How-To](#) | [Permalink](#) | [Comments](#) | [Share](#)

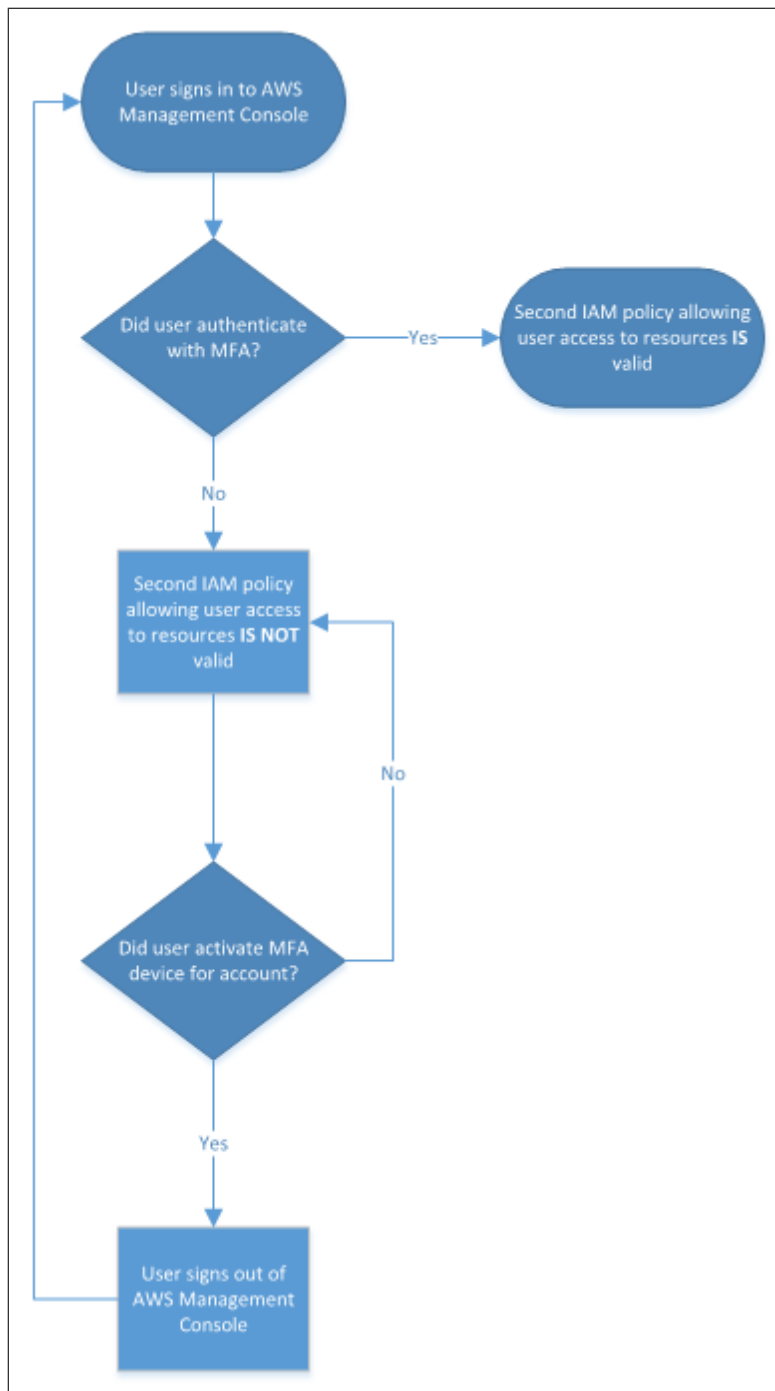
**Note from September 20, 2017:** Based on customer feedback, we have moved the process outlined in this post to the official [AWS documentation](#).

---

AWS Identity and Access Management (IAM) has a list of [best practices](#) that you are encouraged to use. One of those best practices is to [enable multi-factor authentication](#) (MFA) for your AWS root account. MFA verifies your identity through something you know (user name and password) and something you have (MFA hardware or software token).

Enabling MFA for one account is a simple process, and setup on the root account typically only takes a few minutes. But what about large-scale administration of MFA? Centralized provisioning and management can be tedious and scales poorly. Even so, the value of MFA-secured access demands a workable approach for securing your AWS assets.

This post will show you how to grant your users access to provision and manage their own MFA devices while not allowing them access to any AWS resources until they authenticate via their newly provisioned MFA device. The following diagram shows the workflow that this blog post follows.

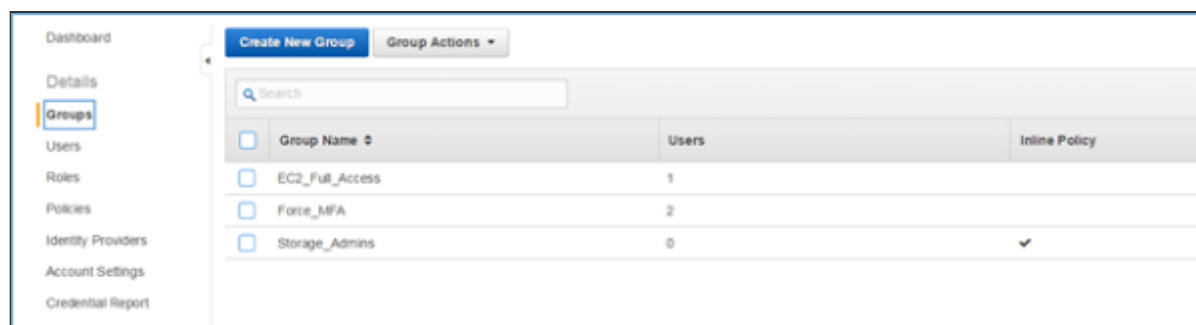


## Overview of this blog post

In this blog post, you will create two IAM groups and then two IAM policies. You will then attach the policies to the groups. After that, you will create test users and add them to the new groups. Finally, you will sign in to the AWS Management Console as one of the newly created users to validate the process. If testing works as expected, on the first sign-in to the console, the test user will only have access to IAM and options for their own user such as resetting their password and managing their own MFA device. Prior to activating and signing in with an MFA device, the users will not have access to anything other than IAM resources. After activating an MFA device, signing out, and signing back in with the MFA device, a user will then have full access to Amazon EC2. (It is important to note that in this post we are using EC2 as an example. You could use any combination of AWS resources, not just EC2.)

## IAM policy types

You can create IAM users and then add them to groups. Then, you can create and attach policies to those groups to control several different AWS features. You can attach [two types of policies](#) to a group. The first is called a [managed policy](#), and the second is called an [inline policy](#). Managed policies are policies that you can attach to multiple users, groups, and roles in your AWS account. In this post, you will use two managed policies: one of the [AWS managed policies](#), and a [customer managed policy](#) that you will create. The following image shows a check mark next to a group that has an inline policy attached to it.

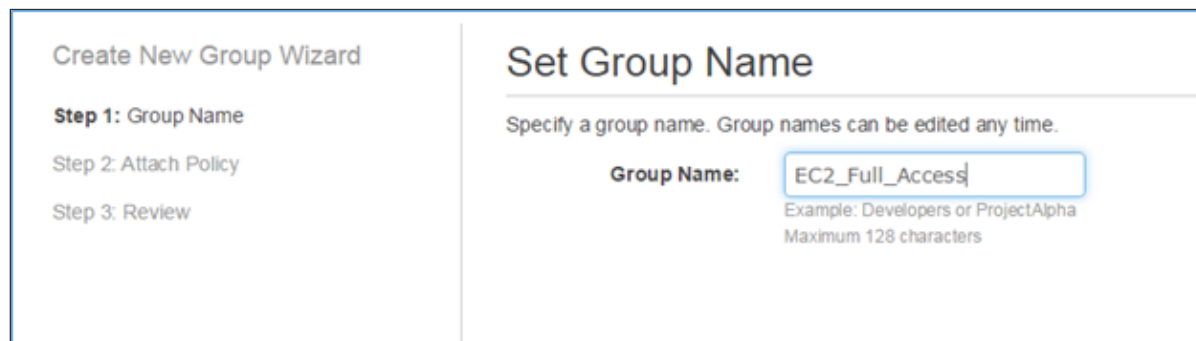


<input type="checkbox"/>	Group Name ↕	Users	Inline Policy
<input type="checkbox"/>	EC2_Full_Access	1	
<input type="checkbox"/>	Force_MFA	2	
<input type="checkbox"/>	Storage_Admins	0	✓

## Create groups

To get started, you will create two example groups. The first group will be called `EC2_Full_Access` and the second group will be called `Force_MFA`. The `EC2_Full_Access` group will use the AWS managed `AmazonEC2FullAccess` policy that grants full access to all EC2 resources. The second group, `Force_MFA`, will have a custom policy attached to it that will grant users access to a subset of IAM privileges, such as managing their own password and enabling an MFA device for themselves. When a user is added to both of these groups, it will have the effect of granting the user EC2 full access only *after* they have successfully authenticated with an MFA device configured for their account.

To get started, go into the **Groups** section of the IAM console, and then click **New Group**. Name the new group `EC2_Full_Access` (as shown in the following image), and then click **Next**.



Create New Group Wizard

**Step 1: Group Name**

Step 2: Attach Policy

Step 3: Review

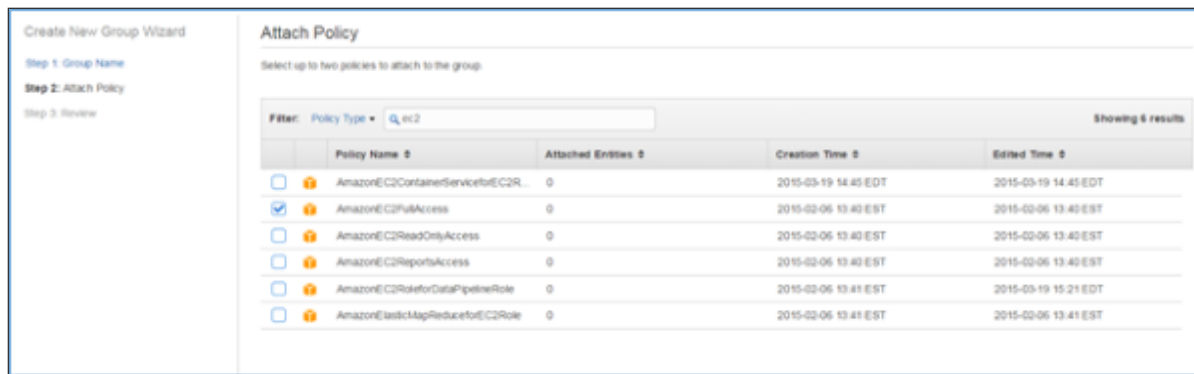
**Set Group Name**

Specify a group name. Group names can be edited any time.

**Group Name:**

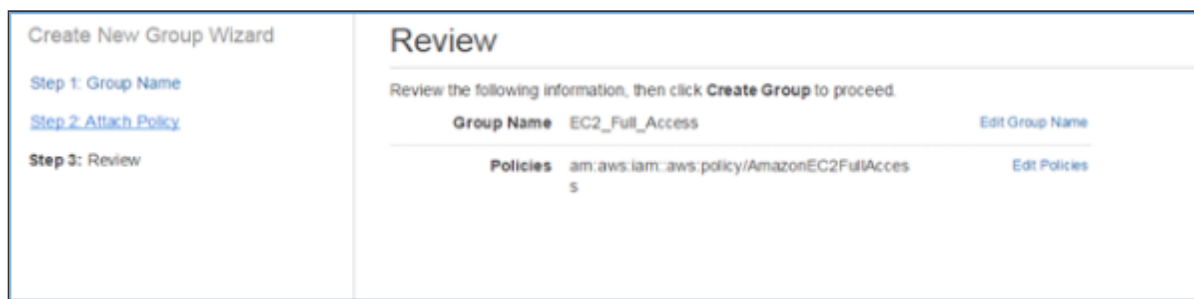
Example: Developers or ProjectAlpha  
Maximum 128 characters

The next page will ask you to attach a policy. Type `ec2` in the **Filter** box to see a list of AWS managed policies related to EC2, as shown in the following image.

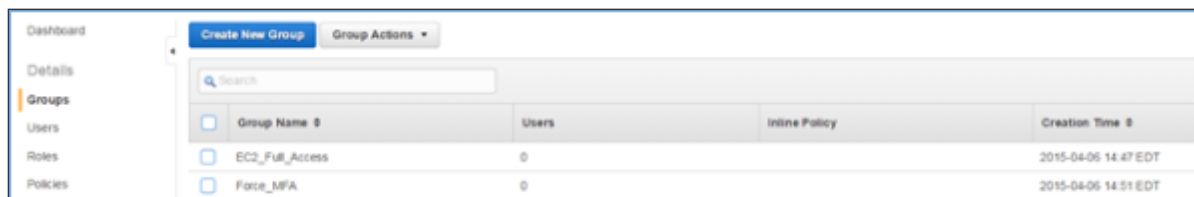


Select the policy called `AmazonEC2FullAccess`, and then click **Next Step**. Make sure the settings you want are in place.

Finally, click **Create Group** to finish creating the `EC2_Full_Access` group, as shown in the following image. You will follow the same steps to create the `Force_MFA` group *except* that you will not select a policy, but will instead create your own policy in steps outlined later in this post.

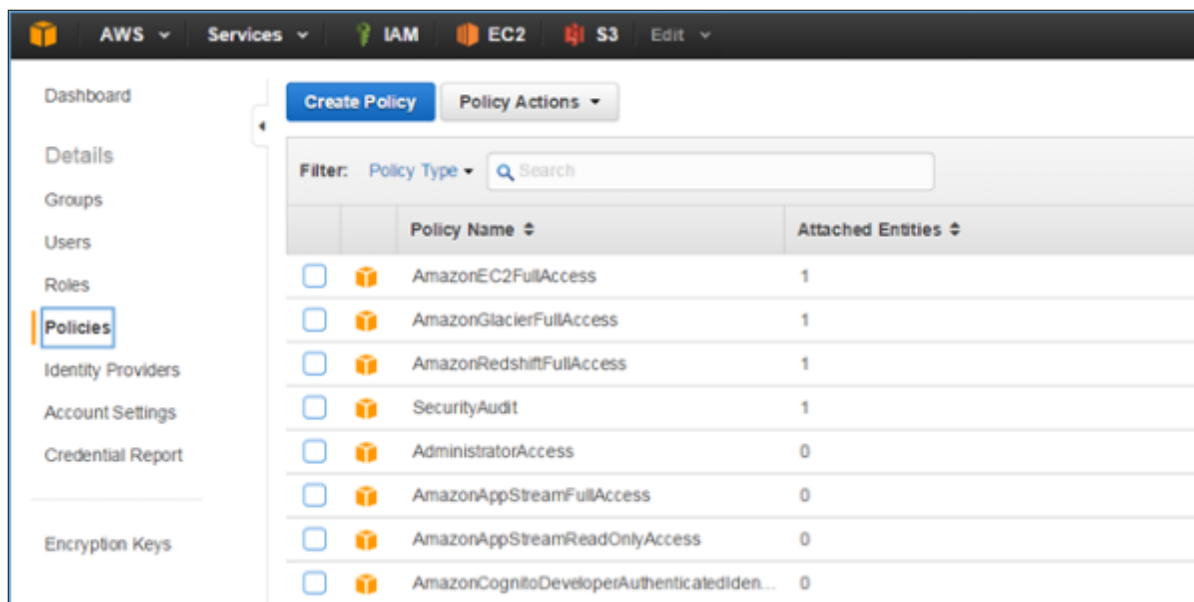


At this point, you should see both new groups in the **Groups** section of the IAM console.

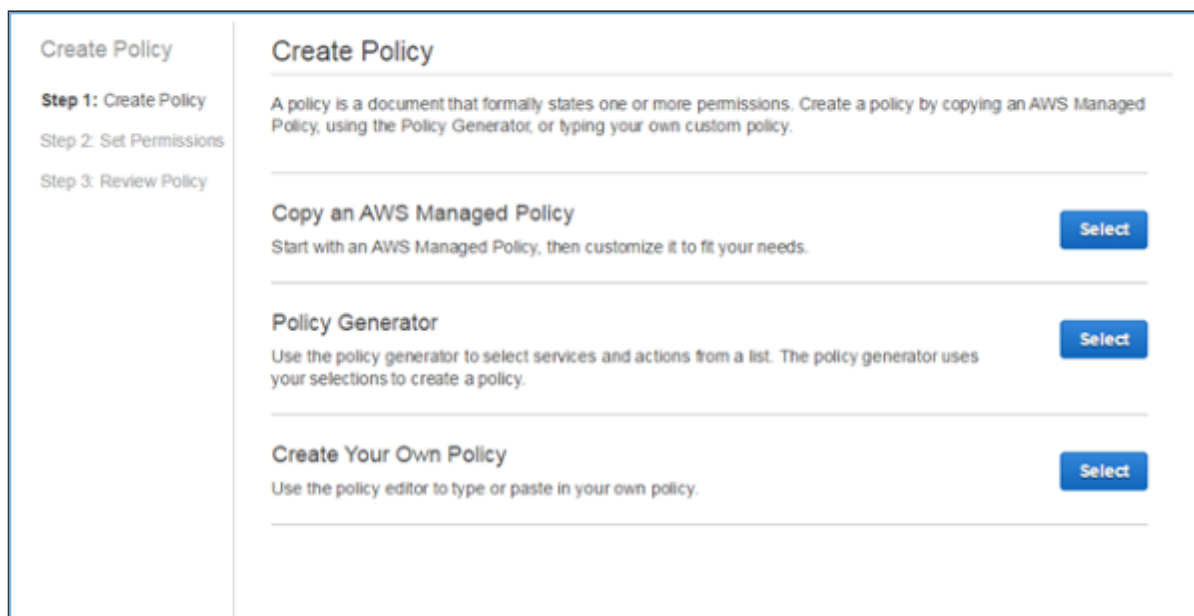


## Create a managed policy

The next steps involve creating a managed policy and configuring it. First, click **Policies**. Then click **Create Policy**.



Note that for **Attached Entities** you will see numbers specific to your account that may not match what is shown in the previous image. Go to the bottom of the page to the **Create Your Own Policy** section, as shown in the following image. Then click **Select**.



Give the policy a meaningful name and description. For this example, call the policy `Force_MFA`, and in the description note, "This policy allows users to manage their own passwords and MFA devices." You can copy and paste [this IAM policy document template](#) into the **Policy Document** section. Be certain to change the places where it says `ACCOUNT-ID-WITHOUT-HYPHENS` with your own AWS account ID. If you do not know your AWS account ID, follow [these steps](#) to find it.

**Create Policy**

Step 1: Create Policy

Step 2: Set Permissions

**Step 3: Review Policy**

### Review Policy

Customize permissions by editing the following policy document. For more information about the access policy language, see [Overview of Policies](#) in the *Using IAM* guide. To test the effects of this policy before applying your changes, use the [IAM Policy Simulator](#).

**Policy Name**

Force\_MFA

**Description**

This policy allows users to manage their own passwords and MFA devices and does not let them do anything else unless they authenticate using MFA

**Policy Document**

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "AllowAllUsersToListAccounts",
6       "Effect": "Allow",
7       "Action": [
8         "iam:ListAccountAliases",
9         "iam:ListUsers"
10      ],
11      "Resource": [
12        "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/*"
13      ]
14    },
15    {
16      "Sid": "AllowIndividualUserToSeeTheirAccountInformation",
17      "Effect": "allow",
```

☒ Use autoforamtting for policy editing

Cancel Validate Policy Previous **Create Policy**

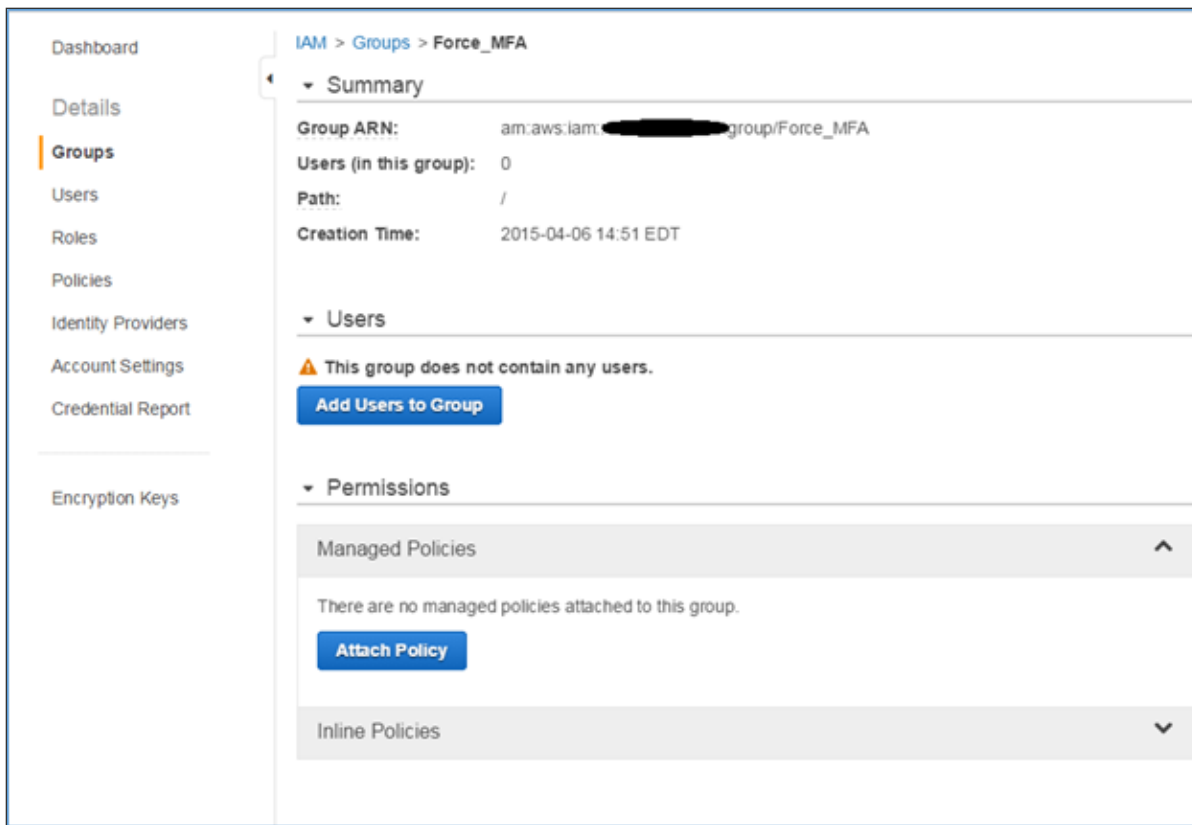
Click the **Validate Policy** button (shown in the previous image) to make sure that there are no errors, and then click **Create Policy**. If you do encounter an error, check that you copied the text correctly from the [IAM policy document template](#).

The policy document is broken into several areas. For more information about what each of these areas does, see the [IAM Policy Elements Reference](#) and [IAM documentation](#).

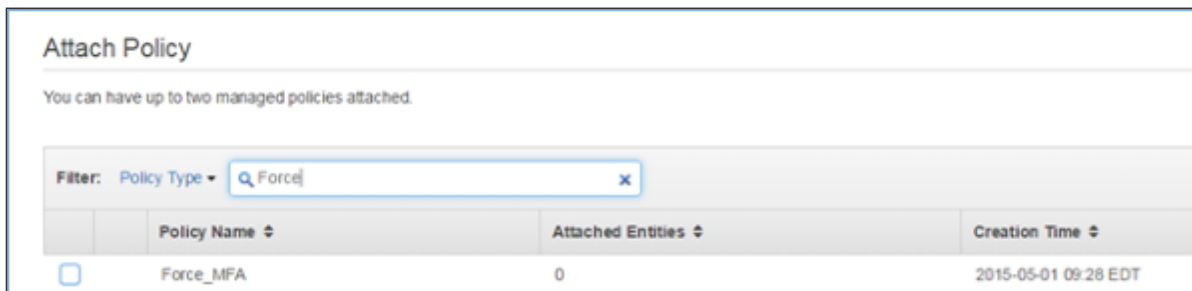
## Attach a customer managed policy

Now that you have the groups and policies set up, you will attach the customer managed policy called `Force_MFA`, which we created previously, to our new custom groups. Then, you can add existing users or create new users, and add them to your groups to test things out.

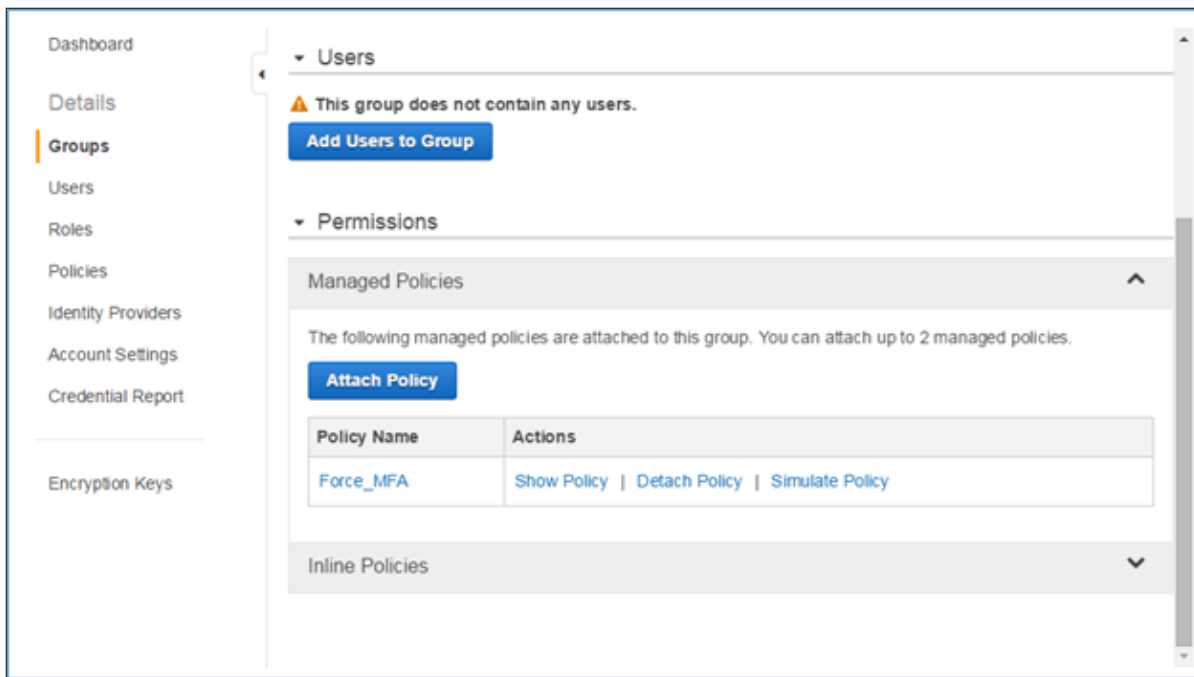
First, go back into the `Force_MFA` group, as shown in the following image.



In the `Force_MFA` group, click **Attach Policy** in the **Managed Policies** section. Type `Force` in the **Filter** box, as shown in the following image.

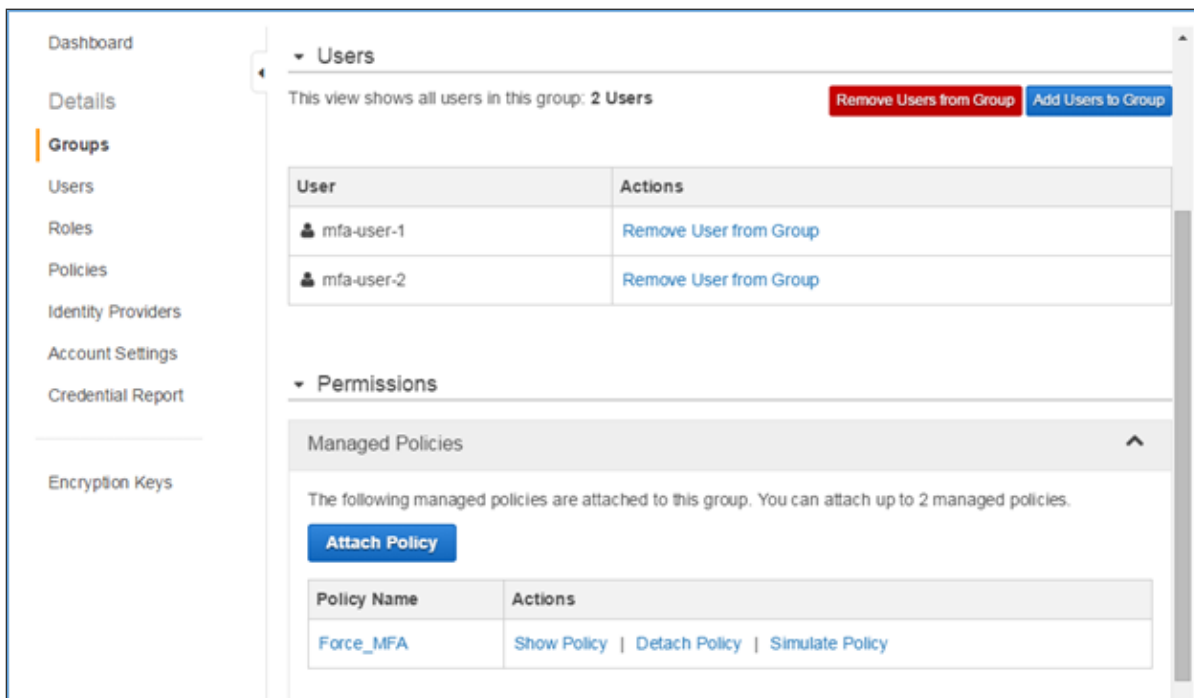


Attach the `Force_MFA` policy we created earlier by clicking **Attach Policy**.



## Create test users

Create test users called `mfa-user-1` and `mfa-user-2`, as shown in the following image. Add them to the `Force_MFA` group first, and then add them to the `EC2_Full_Access` group. You can use your own existing users, or create new ones and add them to the groups. Be sure to add the users to the `Force_MFA` group first so that you do not inadvertently give the user full EC2 access before locking them down with the MFA restrictions. At this point, you should have two test users added to two groups, as shown in the following image.

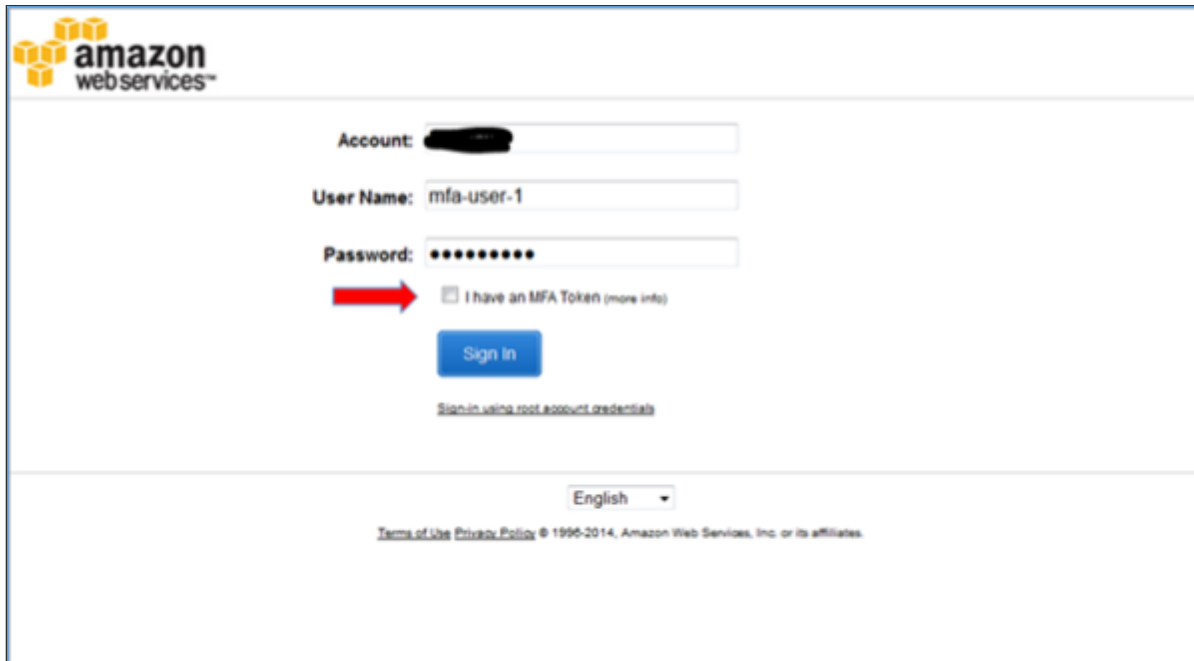


## Make sure it works

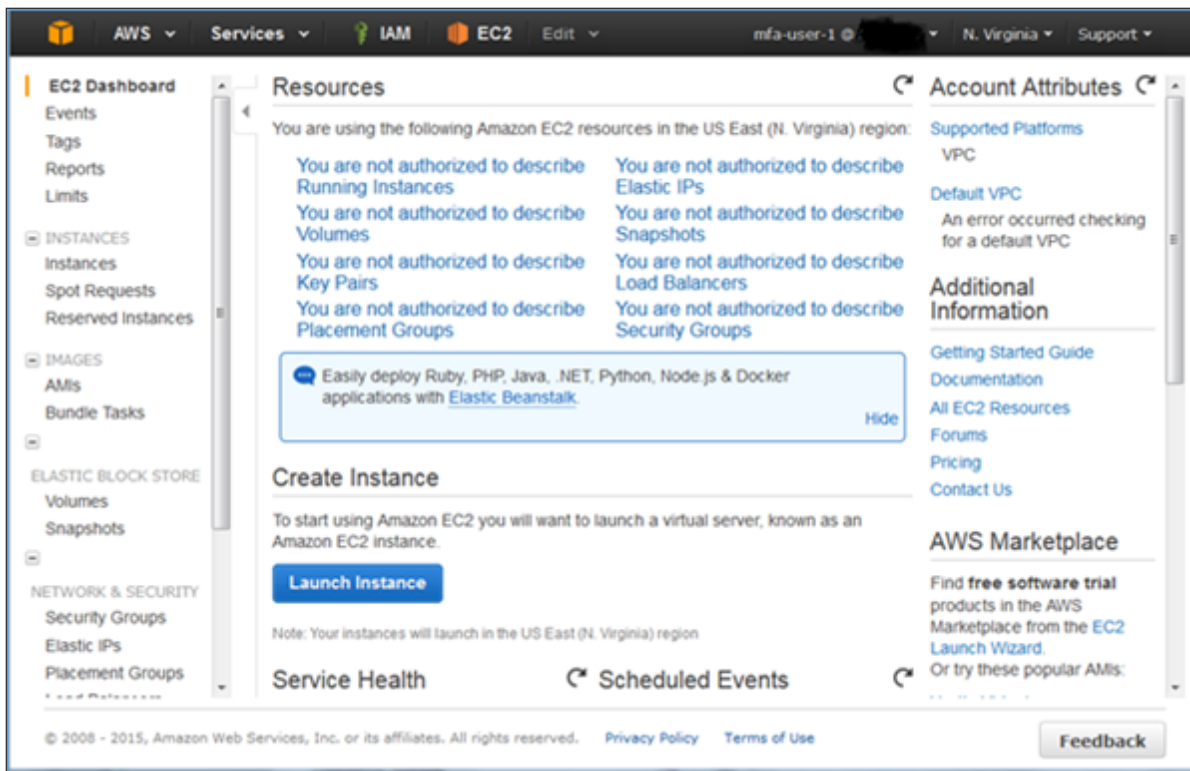


Next, try to sign in as one of these users and provision an MFA device for the user. Browse to the URL you would use for signing in with IAM users. It will generally follow the form of `https://ACCOUNT-ID-WITHOUT-HYPHENS.signin.aws.amazon.com/console`. If you are not sure which sign-in URL to use, you can find it for your account by reviewing [this documentation](#).

On the sign-in page, specify one of the IAM users you added to the two groups earlier. Because this user does not have an MFA device yet, you must clear the **I have an MFA Token** check box, as shown in the following image. You will set up the MFA device later in this post.

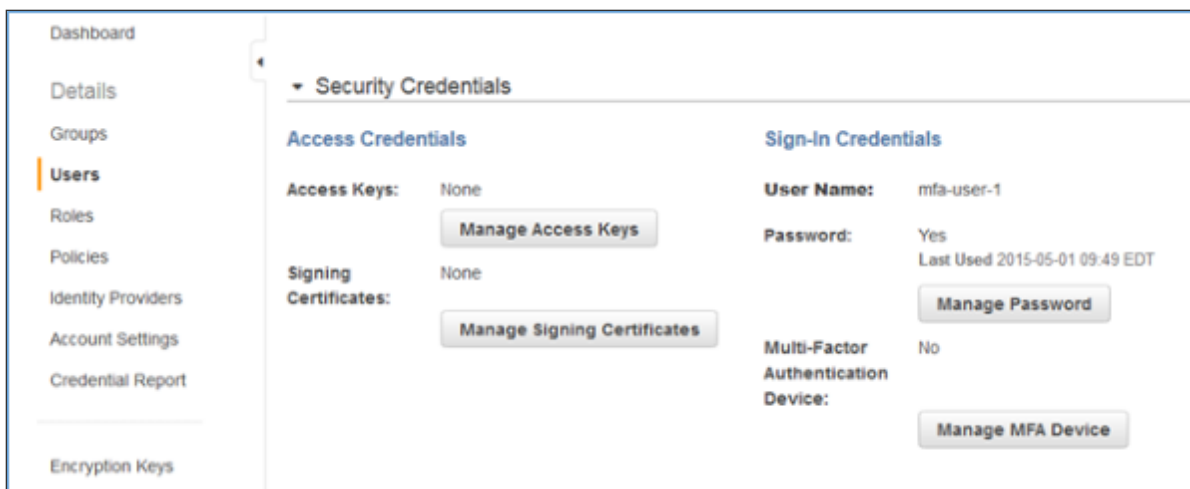


After you have signed in, go to the EC2 Dashboard and note that you should not have access to any resources, as shown in the following image.

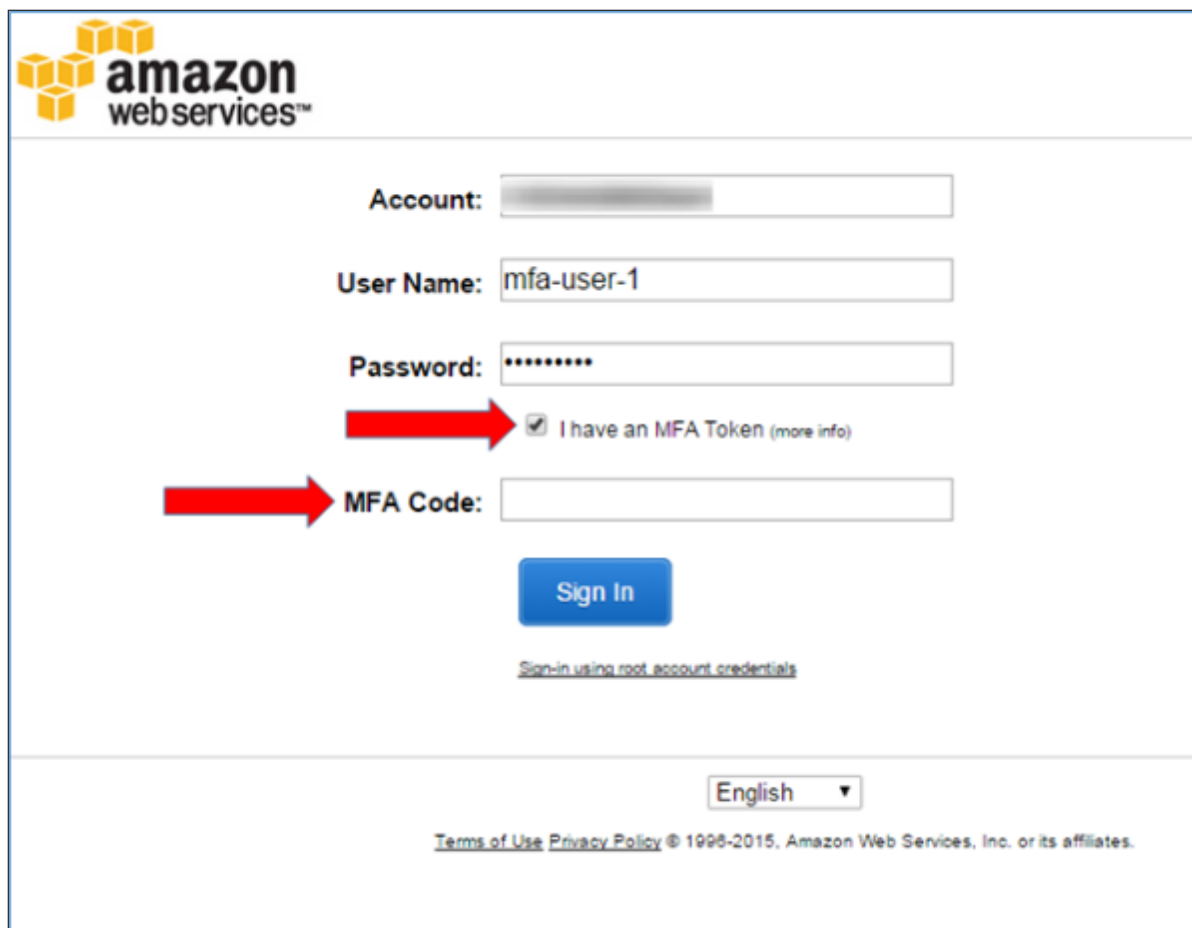


To gain access to the resources, you must first configure an MFA device, and sign in again using the MFA device. To do so, proceed to the IAM console. Note that you will now see several permission “not authorized” errors, which is expected because the user has been granted access to a limited set of IAM resources.

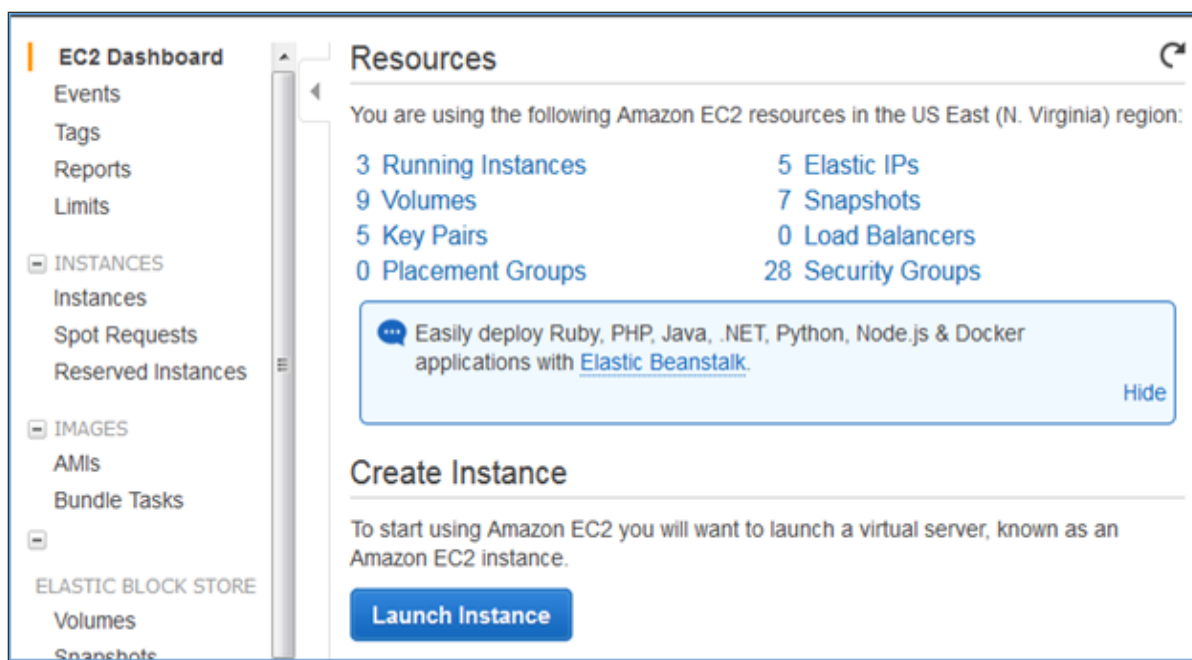
Next, go in to the **Users** section in the IAM console, and then click the user that you used to sign in. Click **Manage MFA Device** (as shown in the following image), and follow the instructions to [configure a virtual or physical MFA device](#). Sign out of the AWS Management Console.



Next, sign back in as the modified user. This time select the **I have an MFA Token** check box, as shown in the following image.



Enter your account information and the MFA code that is displayed on your device. You should now have access to the EC2 Dashboard, which was inaccessible until you configured an MFA device and signed in to the AWS Management Console by using that MFA device.



If you have used IAM and created IAM policies before, you may wonder how we are preventing users from getting access to resources (allowed in the `EC2_Full_Access` policy) until they have configured an MFA device

and authenticated with it. The answer is in the last IAM statement of the `Force_MFA` policy. You should examine it now, and keep in mind the rules IAM uses to allow or deny access to a resource:

- By default, the rule is `deny`. In the `Force_MFA` policy, we explicitly allow the user permission to make some IAM API calls. The user can do nothing else with IAM other than these allowed actions.

```
{
    "Sid": "AllowAllUsersToListAccounts",
    "Effect": "Allow",
    "Action": [
        "iam:ListAccountAliases",
        "iam:ListUsers"
    ],
    "Resource": [
        "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/*"
    ]
},
{
    "Sid": "AllowIndividualUserToSeeTheirAccountInformation",
    "Effect": "Allow",
    "Action": [
        "iam:ChangePassword",
        "iam:CreateLoginProfile",
        "iam>DeleteLoginProfile",
        "iam:GetAccountPasswordPolicy",
        "iam:GetAccountSummary",
        "iam:GetLoginProfile",
        "iam:UpdateLoginProfile"
    ],
    "Resource": [
        "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/${aws:username}"
    ]
},
{
    "Sid": "AllowIndividualUserToListTheirMFA",
    "Effect": "Allow",
    "Action": [
        "iam:ListVirtualMFADevices",
        "iam:ListMFADevices"
    ],
    "Resource": [
        "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:mfa/*",
        "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/${aws:username}"
    ]
}
```

```

    },
    {
      "Sid": "AllowIndividualUserToManageThierMFA",
      "Effect": "Allow",
      "Action": [
        "iam:CreateVirtualMFADevice",
        "iam:DeactivateMFADevice",
        "iam>DeleteVirtualMFADevice",
        "iam:EnableMFADevice",
        "iam:ResyncMFADevice"
      ],
      "Resource": [
        "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:mfa/${aws:username}",
        "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/${aws:username}"
      ]
    }
  ]
}

```

- An explicit `deny` overrides an explicit `allow`. In the `Force_MFA` policy, we explicitly `deny` access to anything other than `"iam:*"` if the user has not logged in with an MFA device (using a `Condition` policy construct, and checking for the absence of the `"aws: MultiFactorAuthAge"` dynamic variable, which is only set when the user logs in using an MFA device). This explicit `deny` is what prevents users from having access to EC2 resources, because it overrides any `allow` statements in all other policies (including `EC2_Full_Access`).

```

{
  "Sid": "DoNotAllowAnythingOtherThanAboveUnlessMFA",
  "Effect": "Deny",
  "NotAction": "iam:*",
  "Resource": "*",
  "Condition": {
    "Null": {
      "aws:MultiFactorAuthAge": "true"
    }
  }
}

```

## Learn more about IAM

This blog post has shown you how to enable your users to provision and manage their own MFA devices while not granting them access to any resources until they have authenticated via their newly provisioned MFA device. By doing this, you take the task of large-scale administration of MFA devices from the job of your administrators and delegate it to your users.

We look forward to hearing how you are using IAM and MFA for your users and the ways we can improve the functionality. You can post comments below, or visit the [IAM forum](#) to post comments and questions about managed policies.

– Mike

**Want more AWS Security how-to content, news, and feature announcements? Follow us on [Twitter](#).**

TAGS: [Amazon EC2](#), [AWS IAM](#), [managed policies](#), [MFA](#)

## Resources

[General Data Protection Regulation \(GDPR\)](#)  
[AWS Cloud Security](#)  
[Amazon GuardDuty](#)  
[Amazon Macie](#)  
[AWS Identity and Access Management \(IAM\)](#)  
[AWS Directory Service](#)  
[AWS Artifact](#)  
[AWS Compliance](#)

---

## Follow

[!\[\]\(73002692dd5e7a64e60946be3158e719\_img.jpg\) Twitter](#)  
[!\[\]\(42837a1907e26cf155e215b5440e265d\_img.jpg\) Facebook](#)  
[!\[\]\(42c4abe8a012119f15571407ccb34aff\_img.jpg\) LinkedIn](#)  
[!\[\]\(211fe2e737577605a46f8d7cec610ff9\_img.jpg\) Twitch](#)  
[!\[\]\(aba70343d9e73e593f58f26a8a807afb\_img.jpg\) Email Updates](#)

## Related Posts

---

[Automate analyzing your permissions using IAM access advisor APIs](#)

[Stream Amazon CloudWatch Logs to a Centralized Account for Audit and Analysis](#)

[Simplify granting access to your AWS resources by using tags on AWS IAM users and roles](#)

[Add Tags to Manage Your AWS IAM Users and Roles](#)

[Restrict Access to your Amazon Connect S3 Bucket](#)

[Restrict access to your AWS Glue Data Catalog with resource-level IAM permissions and resource-based policies](#)

[Use YubiKey security key to sign into AWS Management Console with YubiKey for multi-factor authentication](#)

[AWS Organizations now requires email address verification in order to invite accounts to an organization](#)